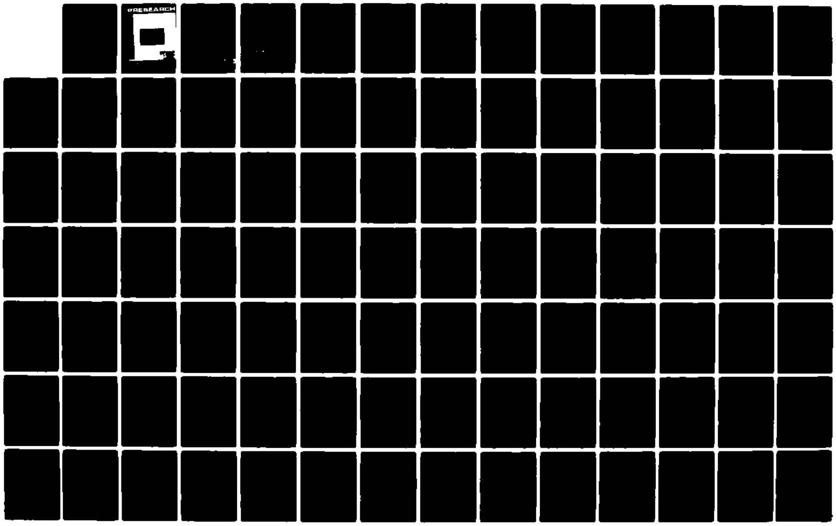
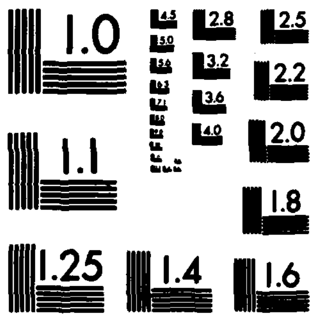
AD-A127 202 FLEET MOORING LEG DESIGN PROGRAM DOCUMENTATION VOLUME 3 1/
SUBROUTINE DESCRIPTIONS(U) PRESEARCH INC ARLINGTON VA
DEC 82 FPO-1-82-(34) N62477-81-C-0025

UNCLASSIFIED F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

FLEET MOORING LEG
DESIGN PROGRAM DOCUMENTATION

Volume 3

SUBROUTINE DESCRIPTIONS

FPO-1-82-(34)

December 1982

General Distribution

PRESEARCH INCORPORATED

1

FLEET MOORING LEG DESIGN PROGRAM DOCUMENTATION

Volume 3

SUBROUTINE DESCRIPTIONS

FPO-1-82-(34)

December 1982

General Distribution

Performed for
Ocean Engineering and Construction Project Office
Chesapeake Division
Naval Facilities Engineering Command
Washington, D.C. 20374
Under
Contract N62477-81-C-0025

This document has been approved for public release and sale; its distribution is unlimited.

DTIC
S ELECTE D
APR 25 1983
E

Presearch Incorporated
2361 South Jefferson Davis Highway
Arlington, Virginia 22202

PRESEARCH INCORPORATED

FLEET MOORING LEG DESIGN PROGRAM DOCUMENTATION

Volume 3

SUBROUTINE DESCRIPTIONS

Accession For

NTIS GRA&I ☒
DTIC TAB ☐
Unannounced ☐
Justification [illegible]

By
Distribution/

Availability Codes

Dist	Avail and/or Special
A	

DTIC COPY INSPECTED 2

III. SUBROUTINE DESCRIPTIONS

MOORΦ1

Main program of the module used to create input parameter data files without invoking the solution procedure. May be used to create input parameter files for a single mooring leg or for a riser for a system of legs.

MOORΦ2

Main program of the module used to obtain a solution for a single basic mooring leg. Calls subroutines that perform the following sequence of functions:

- Query the user for input data
- Solve the mooring leg equations
- Display the solution in tabular form
- Create an elevation view point file (optional)
- Create a plan view point file (optional)

BKDAT

Block data subroutine to initialize certain variables of the common blocks /LUNITS/ and /GCB/. These variables hold the following fixed values:

- logical unit assignments
- standard file suffixes
- size of logical unit 99 (LU99) in bytes
- number of input parameters to be stored in LU99

QUERY

Main subroutine for data input and data conversion. Used by MOORΦ1, MOORΦ2 and MOORΦ4. Permits input parameter values to be entered manually or read from a file, modified by the user, and written to the same file or to a different file. Converts data from text strings to numerical values, and writes those values to a scratch file for later use. When called by MOORΦ4, this subroutine also queries the user for load deflection curve parameters.

GFINIT

Initializes the graphics control block and sets a dash pattern.
Called by two major subsystems:

- The QUERY/ECHO subsystem common to MOORΦ1, MOORΦ2 and MOORΦ4
- The GRAPH1 subsystem used by MOORΦ2 to produce a table of output values.

Not called by the MOORΦ5 program.

RW

Called by QUERY for manual data entry. Reads a 10-character text string from the keyboard and writes it to the internal file LU99 for subsequent modification.

ADDEXT

Concatenates a standard file extension (.VAR, .LDC, .ELV, or .PLN) to the name of a disk file. Any existing extension is overwritten. The standard extension to be used is passed to this subroutine as a parameter.

ECHO

Called by QUERY to display the values of input parameters stored in LU99 for user modification. Permits an unlimited number of value changes and displays. Initial and final values are in the form of character strings; data conversion is performed by a separate subroutine. Writes final string values to a disk file at the user's option.

OUTVAR

Called by ECHO to create an output file of modified input parameter values. This file consists of character strings in LU99, together with the run title and the name of the input file from which the original parameter values were read ("NONE" if the values were keyed in manually). This file is given a name chosen by the user, and may be used as an input parameter file for subsequent runs. If the user elects not to create an output file, then OUTVAR assigns a string of blanks to the output file name; that name, together with the input file name, appears near the top of table Graph 1.

CONVRT

Called by ECHO to convert the text-string values of input parameters in LU99 to numeric values. Also tests the choice of unknown parameters for validity, and terminates the run if that choice is invalid.

RWCOM1

Transfers data between a scratch file on the disk (COMMON.DAT in the library T2DAT.LIB) and a set of eight common blocks that are not used during the solution procedure. This transfer of data is done for the purpose of saving space in main memory. The data include input parameter values, final values for tabular output, elevation view function coefficients, file names, the run title, and the date and time of the run.

HXQRY

Called just prior to the return from QUERY to obtain the current time, which is taken as the starting time for the run. (It is the completion time that appears on table Graph 1, and which is regarded as the "run time".) When used by MOORΦ4, this subroutine also queries the user for curve parameters (range of H, number of points, reference point). When used by MOORΦ2, the subroutine displays the starting time in a message to inform the user that the solution procedure has begun.
This subroutine, when used by MOORΦ4, also performs the function of opening the load deflection curve point file and writing the graph type, title, date and time to that file.

SOLVE

Main subroutine for the solution procedure of MOORØ2.
Calls, in sequence, the following subroutines:

- Preprocessor subroutine PRSLV
- Solution subroutine:
  - TAUT if simple leg, taut line solution
  - SLACK if simple leg, slack line solution
  - CSLACK if compound leg
- Postprocessor subroutine:
  - EPSLV if simple leg
  - CEPSLV if compound leg

PRSLV

Main subroutine for the preprocessing of input data. Performs the following functions:

- Initializes all variables in the common block /VGLOB/ to zero. This common block holds double precision values for all input variables and all essential output variables generated by the solution procedure.
- Sets the values of standard constants.
- Computes ocean floor variables from point coordinates. These variables include the maximum floor slope, the direction of maximum slope, and the water depth at the origin. They also include certain fixed working variables that are determined by the ocean floor and the anchor separation.
- Reads the values of input parameters into double precision variables of /VGLOB/ with the necessary unit conversions (eg, kips → pounds for force).
- Tests each branch of the leg for the presence of negative weights.
- Determines the array indices of unknown variables.

SECNT

Computes the trigonometric secant of an angle, given its tangent.

NWGT

Function subroutine called by PRSLV to determine whether an array containing the parameters for a simple leg, or one branch of a compound leg, specifies any components having negative weight. Returns the value 1 if there is a negative weight component, and returns ϕ otherwise.

UMAP

Function subroutine called by PRSLV to determine the array index of an unknown input parameter for a simple leg, given the position of that parameter in the internal file LU99. during the QUERY/ECHO procedure.

CUMAP

Function subroutine called by PRSLV to determine the array index of an unknown input parameter for a compound leg, given the position of that parameter in the internal file LU99 during the QUERY/ECHO procedure.

ISORT

Called by PRSLV to sort the array indices of unknown input parameters into increasing order.

TAUT

Called by SOLVE to carry out the taut line solution procedure for a simple leg. Determines the type of iterative algorithm to be used (one-dimensional secant method or two-dimensional Steffensen's method) from the choice of unknown input parameters. Obtains the solution by calling the subroutine for the appropriate algorithm.

STEF2A

Called by TAUT to obtain a taut line solution for a simple leg when the buoy displacement is known. Queries the user for an initial guess for each of the two unknown parameters, then iterates to a solution using the two-dimensional Steffensen's method.

CHS2A

Called by STEF2A to query for initial values for a two-dimension
Steffensen iteration. Converts values from the standard units
for input to the units in which the solution is carried out.

SEC1A

Called by TAUT to obtain a taut line solution for a simple leg when the buoy displacement is unknown. Queries the user for two initial guesses for the other unknown parameter, then iterates to a solution using the one-dimensional secant method.

CHS1A

Called by SEC1A to query for initial values for a one-dimensional secant iteration. Converts values from the standard units for input to the units in which the solution is carried out.

CALC1

Fundamental calculation subroutine for the taut line solution procedure. Computes catenary slopes at the endpoints of chain segments (nodes) and the horizontal and vertical displacement of each chain segment. Input data for CALC1 consist of values for the hardware components, the horizontal load, the catenary angle at the anchor, and the number of chain segments in the leg.

SLACK

Called by SOLVE to carry out the slack line solution procedure for a simple leg. Determines the type of iterative algorithm to be used (one-dimensional secant method or two-dimensional Steffensen's method) from the choice of known input parameters (H, φ_H) or (R_{TOT}, φ_R) or (X_{TOT}, Z_{TOT}). Obtains the solution by calling the subroutine for the appropriate algorithm

STEF2V

Called by SLACK to obtain a slack line solution for a simple leg when (R_{TOT}, φ_R) or (X_{TOT}, Z_{TOT}) has been specified as choice of known input parameters. Carries out a two-dimensional Steffensen iteration over the vertical and horizontal components of the tension at the buoy (denoted "vertical tension" and "horizontal tension", respectively). This subroutine also generates the initial values for the iteration, so that the user is not required to input a guess.

SEC1V

Called by SLACK to obtain a slack line solution for a simple leg when (H, φ_H) has been specified as choice of known input parameters. Carries out a one-dimensional secant iteration over the vertical tension, generating the initial value for that iteration so that the user need not input a guess. Also called by STEF2V in the process of generating the initial estimate for a two-dimensional Steffensen iteration.

VCRITΦ

Computes values for an array VCΦ of six elements, where VCΦ(i) equals the weight of that part of a simple leg lying between node i and the buoy. These values are used to compute vertical tension at the buoy at critical points in the behavior of the leg. These critical points mark the beginning or end of a chain segment's rise above the floor as the tension on the buoy is increased.

ESTV

Computes the vertical tension for a simple leg having no sinkers and only one weight of chain. Vertical tension is computed from the horizontal load, the depth of water at the anchor, the ocean floor slope, and the scope and linear weight of the leg. The simple leg to which this subroutine is applied serves as a model for a more complicated simple leg. The value that is returned for vertical tension is used as an initial value for the one-dimensional iteration to a slack line solution for the leg.

LENS

Function subroutine that returns the slack length of chain for a simple leg having no sinkers and only one weight of chain. This is computed from the horizontal load, the depth of water at the anchor, the ocean floor slope, and the scope and linear weight of the leg. Called by ESTV to obtain vertical tension for such a leg. Also applied to the riser of a compound mooring leg when the junction of the leg lies on the ocean floor.

VFUN

Function subroutine that returns a value for vertical tension that is algebraically larger than the value that is passed to it as a parameter. Used to generate a second initial value for the one-dimensional iteration to a slack line solution for a simple leg. Also used to adjust the value of vertical tension when it goes out of bounds during a two-dimensional iteration.

CALC2

Fundamental calculation subroutine for the slack line solution procedure. Computes the slack length of chain, catenary slopes at the endpoints of chain segments, and the horizontal and vertical displacement of each chain segment. Input data for CALC2 consist of the number of chain segments in the leg, values for all hardware parameters, the ocean floor slope, the horizontal load, and the vertical tension.

EPSLV

Main postprocessing subroutine for the simple leg solution. Performs the following functions:

- Initializes all output variables to the value 9999.99, which is automatically suppressed when Graph 1 is printed. Thus, only the output variables pertaining to the particular leg will be printed.
- Reads solutions for the unknown input parameters to the corresponding input variables. Values of the input parameters are printed in the upper left part of Graph 1. For the slack line solution, the angle between the chain and the ocean floor at the anchor is always included as an unknown input parameter.
- Computes catenary function coefficients for the elevation view.
- Computes angles and tensions at the nodes of the leg.
- Computes node coordinates
- Reads all pertinent output values to the appropriate variables for Graph 1, with unit conversion as necessary. Values are converted from the units used by the solution procedure to the standard input/output units (eg, pounds → kips for force).

RDBACK

Called by EPSLV to read the values of unknown input parameters to the appropriate output variables, with unit conversion as necessary.

ELV1

Called by EPSLV to compute parameters for the elevation view. These parameters include catenary function coefficients; the horizontal compression factor; and projected values for the slack length, horizontal displacements and ocean floor slope. Only the function coefficients are used by the current version of the elevation view program.

GCOEFF

Computes function coefficients for the catenaries that comprise a simple leg, to be used in drawing an elevation view of the leg. Coefficients for each catenary are given with respect to a coordinate system local to that catenary. This coordinate system has its x-axis through the anchor and its y-axis through that endpoint of the catenary that is closest to the anchor.

CTEN1

Called by EPSLV to compute angles and tensions at nodes of a simple leg. Angles are expressed in degrees and tensions are expressed in kips.

CSLACK

Called by SOLVE to carry out the compound leg solution procedure. Calls the appropriate subroutines according to the type of leg (spider plate vs equalizer) and the choice of known input parameters.

CPREPΦ

Sets the values of certain variables that are held fixed during the compound leg solution procedure. These variables include the following:

- Array indices for catenary slopes at the junction
- Array indices for the uppermost scope of each branch
- Maximum and minimum values for the uppermost scope of each branch (which vary during the equalizer solution procedure)
- Sum of the uppermost scopes of the two branches
- Initial slippage of equalizer
- Initial value for uppermost scope of branch A

CPREP1

Sets values of certain variables that remain fixed during a spider plate solution procedure. These variables include the following:

- Total scope and weight for each branch
- Critical tensions for each branch
- An index to indicate whether geometric considerations permit the branches to form a triangle on the ocean floor
- Horizontal branch directions, and junction coordinates, for the case in which the branches form a triangle on the ocean floor.

SUMSC

Computes the total scope and weight for one branch of a compound leg.

CPREP2

Sets values of compound leg solution variables that depend only on the horizontal load. These variables hold trigonometric functions of the load direction and of the effective ocean floor angle in that direction. They remain fixed during any solution procedure for which the load direction is known.

CPREP3

Sets the values of compound leg solution variables that depend on the horizontal load vector and the riser hardware parameters. These variables remain fixed during any solution procedure for which the magnitude and direction of the load is known.

CSSHP

Main subroutine for the spider plate solution when the magnitude and direction of the load are both known. Carries out a series of tests to determine which of four possible types of configuration the leg assumes, then calls the appropriate subroutines to obtain a solution.

PHIAB

Computes horizontal branch directions for a compound leg, given anchor separation and horizontal branch displacements. Used by spider plate solution procedure when both branches are under tension.

HSPLIT

Computes horizontal branch loads, given the vector load on the buoy and the horizontal branch directions. Also computes effective ocean floor slopes in the branch directions. Used by the spider plate solution procedure when both branches are under tension.

TRISR

Computes the catenary slope at the buoy end of a compound leg riser and the vertical displacement of the riser, given the slope at the junction end of the riser. Used by spider plate solution procedure when the riser has zero slack length; this can be true when the junction lies either on or above the ocean floor. This subroutone is called whenever it is desired to compute a buoy elevation independently of the water level.

SCA7A8

Computes the trigonometric secants of the angles at the ends of a compound leg riser, given the tangents of those angles. Used by the spider plate solution procedure to compute the vertical displacement of a riser.

SRISR

Computes the slack length, catenary slopes and displacements for a compound leg riser, given the y-coordinate of the junction end. Used by the spider plate solution procedure when the junction lies on the ocean floor. This subroutine is called when the buoy elevation is assumed to equal the water level.

TAN1

Function subroutine to compute the tangent of the algebraically smaller of two angles, given the differences between their tangents and their secants. Called by subroutine SRISR of the spider plate solution procedure when the slack length of the riser is zero.

X4CALC

Computes the horizontal displacement of the catenary portion of a compound leg riser, given the tangents and secants of the angles at the ends of the riser. Used by the spider plate solution procedure.

TSPLIT

Computes the tension at the junction end of each branch of a compound leg when the junction lies on the ocean floor and both branches are under tension. Called by spider plate solution procedure; the tensions are saved for use in the equalizer solution.

JTEN

Computes the force exerted by a compound leg junction on the leg branches when the junction lies on the ocean floor. A single force is returned. One or both branches may be under tension. Called by spider plate solution procedure. The force computed by this subroutine is used to compute branch tensions for the equalizer solution procedure.

JUNCT

Computes the contact angle for a compound leg equalizer when both branches of the leg are under tension. The junction may lie on or above the ocean floor. If the junction lies above the ocean floor, this subroutine also computes the tension at the end of each branch; those tensions are also used by the equalizer solution. Called by the spider plate solution procedure.

EBUOY

Preliminary subroutine to obtain spider plate solution when only one branch is assumed to be under tension. Initializes variables for tension branch parameters. Computes buoy elevation under the assumption that the junction is just touching the ocean floor, with none of its weight supported by the floor. The computed buoy elevation is used to determine whether a solution having the junction on the floor is possible.

SCOIL

Called by spider plate solution procedure to obtain the solution for a configuration having the junction above the ocean floor and only one branch under tension. Carries out a one-dimensional iteration, treating buoy elevation as a function of the vertical tension for the tension branch. This subroutine also computes the length of the coiled part of the coil branch.

XSECV

Carries out an iterative solution for an equation of the form $f(V) = V_{KNOWN}$ arising in the spider plate solution procedure, where V denotes vertical tension for a branch of the leg and f may be one of four functions:

1. $f(V) =$ horizontal displacement of branch
2. $f(V) =$ vertical displacement of branch
3. $f(V) =$ vertical distance from junction to ocean floor
4. $f(V) =$ buoy elevation

The iteration is one-dimensional and employs bisection and two versions of the secant method. The technique used at any stage of the iteration is determined by the type of function and the behavior of the test values.

SHIFT

Transfers values among elements of each of four arrays used by the one-dimensional iteration conducted by XSECV. These arrays have three elements each, denoting the current and previous two estimates for test variables. The subroutine accepts a pair of array indices i, j and sets element i equal to element j for each array.

SUBVX

Computes function values for the one-dimensional iteration conducted by XSECV. Accepts two array indices. The first indicates which estimate is involved (current or one of two previous estimates). The second denotes the type of function (1, 2, 3, or 4).

WGTH

Computes the weight of the hanging part of the coil branch of a compound leg when the junction lies above the ocean floor and only one branch is under tension. This weight is added to the weight of the junction for the computation of riser angles and displacements. Used by spider plate solution procedure.

CERISR

Computes catenary slopes and vertical displacement for a compound leg riser, as well as buoy elevation, when the junction lies above the ocean floor and only one branch is under tension. Used by spider plate solution procedure.

SECVIT

Carries out a one-dimensional secant iteration, with no validity checks or corrections applied to test variables, for the type of equation for which XSECV is designed. Called by XSECV whenever it has been determined that test values are sufficiently close to the true solution.

STEFAB

Called by spider plate solution procedure to obtain the solution for a configuration having both branches under tension and the junction above the ocean floor. Carries out a two-dimensional iteration (Steffensen's method) to solve the following system of equations:

$$\begin{cases} \Delta y_{JUNCT}(x_A, x_B) = 0 \\ \Delta y_{BUOY}(x_A, x_B) = 0 \end{cases}$$

where x_A and x_B are total horizontal branch displacement

Δy_{JUNCT} is the difference between computed junction elevations

Δy_{BUOY} is the difference between the water level and the computed buoy elevation

CALC3

Carries out the basic calculation for the iteration conducted by STEFAB. Accepts values for branch displacements x_A and x_B. Returns values for elevation differences Δy_{JUNCT} and Δy_{BUOY}.

EDGPT

Called by STEFAB to adjust invalid points (x_A, x_B) that are generated by the Steffensen iteration. Determines the intersection of two curves:

- the line segment joining the invalid point with the most recent valid point
- one of the two hyperbolic curves that bound the region of validity for points (x_A, x_B) representing a pair of branch displacements.

STEFAB then selects a point on the line segment that lies in the interior of the valid region. The iteration resumes with this interior point, which represents movement in the direction originally indicated.

CSEHP

Main subroutine for the equalizer solution when the magnitude and direction of the horizontal load are known. Obtains the solution by an iterative procedure that regards an equalizer as a movable spider plate. This procedure calls the spider plate subroutine CSSHP for different positions of the junction and compares the tensions at the branch ends.

FTEN

Computes the value of the test variable for the equalizer solution procedure conducted by CSEHP. This value is a function of the branch tensions, the contact angle at the equalizer, and the coefficient of friction along the equalizer. If the equalizer slips from its initial position, but does not slip the entire length of the chain, then the value returned by FTEN for the final position must be zero.

CSSPR

Main subroutine for the spider plate solution when the known input parameters are load direction and buoy displacement. Obtains the solution by an iterative procedure that tests different values of load magnitude H and compares the computed value of buoy displacement R_{TOT} with the known value. The spider plate solution subroutine CSSHP is called once for each iteration.

CSEPR

Main subroutine for the equalizer solution when the known input parameters are load direction and buoy displacement. Obtains the solution by an iterative procedure that is exactly analogous to that of CSSPR. The equalizer solution procedure CSEHP is called once for each iteration.

CSSXZ1

First of two main subroutines for the spider plate solution when the buoy coordinates (X_{TOT}, Z_{TOT}) are the choice of known input parameters. Returns an estimate for the X and Z components (H_X, H_Z) of the horizontal load. This estimate then serves as the initial value for a two-dimensional iterative process carried out by CSSXZ2 to obtain the exact solution for the load vector. The estimate is obtained by calling CSSPR with load direction and buoy displacement corresponding to the vector (X_{TOT}, Z_{TOT}).

CSSXZ2

Second of two main subroutines for the spider plate solution when the buoy coordinates (X_{TOT}, Z_{TOT}) are the choice of known input parameter. Conducts a two-dimensional Steffensen iteration that treats (X_{TOT}, Z_{TOT}) as a function of (H_X, H_Z), with initial value generated by CSSXZ1.

CSEXZ1

First of two main subroutines for the equalizer solution when the buoy coordinates (x_{TOT}, Z_{TOT}) are the choice of known input parameters. Exactly analogous to CSSXZ1.

CSEXZ2

Second of two main subroutines for the spider plate solution when the buoy coordinates (X_{TOT}, Z_{TOT}) are the choice of known input parameters. Exactly analogous to CSSXZ2.

CEPSLV

Main postprocessing subroutine for the simple leg solution.
Performs the following functions:

- Initializes all output variables to the value 9999.99, which is automatically suppressed when Graph 1 is printed. Thus, only the output variables pertaining to the particular leg will be printed.
- Computes the angles between the chain and the ocean floor at the anchor and assigns their values to the appropriate input variables. Values of the input parameters are printed in the upper left part of Graph 1.
- Computes catenary function coefficients for the elevation view.
- Computes water depth at each anchor.
- Computes coil length, slack lengths, branch directions and branch loads as necessary. (The solution procedure for a given type of configuration will compute some, but not all, of these values.)
- Computes branch component displacements and slopes as necessary.
- Computes angles and tensions at nodes of the leg.
- Computes node coordinates.
- Reads all pertinent output values to the appropriate variables for Graph 1, with unit conversion as necessary. Values are converted from the units used by the solution procedure to the standard input/output units (eg, pounds → kips for force).

CRDBAK

Called by CEPSLV to compute angles between the chain and the ocean floor at the anchors and to assign their values to the appropriate input variables. Originally designed to handle any selection of unknown input variables.

COMMAP

Function subroutine called by CRDBAK to map array elements to input variables, for input parameters selected as unknown. In the current version of the program, the only input parameters that are involved are the angles between the chain and the ocean floor at the anchors.

CONTYP

Function subroutine called by CRDBAK to determine the type of unit conversion, if any, for a given unknown input parameter.

FNOD

Called by CEPSLV to compute branch component displacements and angles when an entire branch of a compound leg lies on the ocean floor. It is designed to handle the case in which the junction end of the branch is collapsed into a single point.

LENH

Computes the length of the hanging part of the coil branch of a compound leg, given its weight, when the junction lies above the ocean floor and only one branch is under tension. Called by CEPSLV to adjust vertical displacements for the components of such a branch.

ELV2

Called by CEPSLV to compute parameters for the elevation view. These parameters include catenary function coefficients; horizontal compression factors; and projected values for slack lengths, horizontal displacements and ocean floor slopes. Only the function coefficients are used by the current version of the elevation view program.

CTEN2

Called by CEPSLV to compute angles and tensions at the nodes of the coil branch of a compound leg. Used when the junction lies above the ocean floor and only one branch is under tension. Angles are expressed in degrees and tensions are expressed in kips.

CTEN3

Called by CEPSLV to compute angles and tensions at the nodes of a compound leg. Not used for the coil branch of a configuration having the junction above the ocean floor and only one branch under tension. Angles are expressed in degrees and tensions are expressed in kips.

GRAPH1

Main subroutine for printing the table of output values (Graph 1) produced by MOORΦ2. Consists of a sequence of calls to subroutines that write individual sections of the table.

GRIN1

Called by GRAPH1 to print the header, legend of units, and first part of the input parameter list of Graph 1. The last line printed begins with the following descriptor: "LINEAR WEIGHT OF SEGMENT 2".

GRIN2

Called by GRAPH1 to complete the printing of the input parameter list. The first line printed begins with the following descriptor: "WEIGHT OF SINKER 2".

GROUT1

Called by GRAPH1 to print solutions for the unknown input parameters. For the taut line solution, the unknown parameters consist of the two input parameters explicitly chosen as unknown. For the slack line solution, the unknown parameters consist of:

- buoy displacement if load magnitude and direction were given
- load magnitude if buoy displacement and load direction were given
- load magnitude and direction if buoy coordinates were given.

GROUT2

Called by GRAPH1 to print computed output values. These consist of the following:

- ocean floor angles
- branch directions
- length of chain coiled on the floor
- slack lengths of chain
- branch loads
- node coordinates
- angles and tensions at the nodes

This subroutine prints everything associated with the table of output values. This includes the symbols identifying nodes (columns) and types of output variables (rows), as well as the table of descriptors for row headings.

GRAPHS

Called by GRAPH1 to draw the stick figure. After doing so, this subroutine waits for function key input to erase the screen and continue the program MOORΦ2. All graphic work performed by this subroutine is localized in a single call to subroutine STICK.

STICK

Called by GRAPHS to draw the stick figure of Graph 1.

SINKER

Called by STICK to draw sinker symbols (open triangles) on the stick figure of Graph 1.

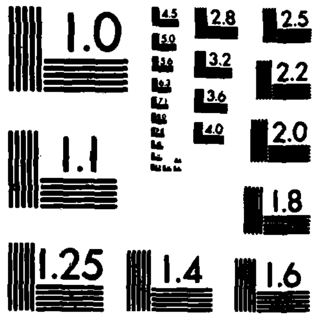
ANCHOR

Called by STICK to draw anchor symbols (figure X) on the stick figure of Graph 1.

AD-A127 202 FLEET MOORING LEG DESIGN PROGRAM DOCUMENTATION VOLUME 3 2/2
SUBROUTINE DESCRIPTIONS(U) PRESEARCH INC ARLINGTON VA
DEC 82 FPO-1-82-(34) N62477-81-C-0025

UNCLASSIFIED F/G 9/2 NL

END
DATE
FILMED
5 83
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ELIZER

Called by STICK to draw an elizer symbol (closed triangle) to represent the junction on the stick figure of Graph 1.

BUOY

Called by STICK to draw a buoy symbol (open square) on the stick figure of Graph 1.

ELVPNT

Main subroutine for production of an elevation view point file by MOORΦ2. Asks the user whether such a file is desired. If so, then ELVPNT computes the coordinates and symbol codes of the elevation view points, and determines the minimum and maximum value for each coordinate axis on the graph. These data, together with the date, time and title of the run, are written to a disk file in the proper format for program MOORΦ5.

SYMSNK

Called by ELVPNT to determine the proper symbol for a sinker. This symbol depends on the sign of the sinker's weight: positive, negative, or zero.

WELVPT

Called by ELVPNT to compute the coordinates of an elevation view point, given the coordinates of the corresponding point on the mooring leg in the local coordinate system used by ELVPNT. After computing the point's coordinates, WELVPT writes them to the output disk file, together with the symbol code for that point, and updates the point counter.

ELVCAT

Called by ELVPNT to compute the elevation view coordinates and symbol codes for points along a single catenary, given the function coefficients for that catenary.

PLNPNT

Main subroutine for the production of a plan view point file by MOORØ2. Asks the user whether such a file is desired. If so, then PLNPNT computes the coordinates and symbol codes of the plan view points, and determines the minimum and maximum value for each coordinate axis on the graph. These data, together with the date, time, and title of the run, are written to a disk file in the proper format for program MOORØ5.

PLNSNK

Called by PLNPNT to determine the proper symbol for a sinker. Identical to SYMSNK except for the declaration of common block /VELVPT/.

WPLNPT

Called by PLNPNT to compute the coordinates of a plan view point, given the coordinates of the corresponding point on the mooring leg in the local coordinate system used by PLNPNT. After computing the point's coordinates, WPLNPT writes them to the output disk file, together with the symbol code for that point, and updates the point counter.

PLNCAT

Called by PLNPNT to compute the plan view coordinates and symbol code for the point at the upper end of a catenary.

MOORФ4

Main program of the module used to create a point file for a load deflection curve. Calls subroutines that perform the following sequence of functions:

- Query the user for an input data file, permitting the echo and modification of input data. Query for the curve parameters: range of load, number of points, reference point. Open the output file and write the graph type, run title, and date and time of the run to the file.
- Preprocess the solution variables.
- Compute the maximum buoy displacement, and write the maximum and minimum values for curve coordinates to the output file. For each distinct value of the load H, compute the corresponding buoy displacement X and write the curve coordinates (X, H) to the output file.

The point file produced by this program is in the proper format to be read and graphed by program MOORФ5.

HXCALC

Main subroutine for computing load deflection curve point coordinates and writing them to the point file. Consists of subroutine calls determined by leg type.

HXCLC1

Called by HXCALC to compute load deflection curve point coordinates for a simple leg and write them to the point file. Also writes the minimum and maximum values of the curve coordinates to the file. If the minimum value of the load is H=0, then the corresponding displacement is determined geometrically.

HXCLC2

Called by HXCALC to compute load deflection curve point coordinates for a compound leg and write them to the point file. Also writes the minimum and maximum values of the curve coordinates to the file. If the minimum value of the load is H=0, then the corresponding displacement is determined by extrapolation from two small values of H.

CSXHP

Called by HXCLC2 to carry out the solution for a compound leg. Each call to CSXHP returns the buoy displacement for a single point on the curve. Accepts a parameter that indicates whether an equalizer solution will use the equalizer position of the previous point as the initial value when iterating to the solution for the current point.

MOORΦ5

Main program for graphing an elevation view, plan view, or load deflection curve. Reads all data for the graph from a disk file of the user's choice. The graph includes the date and time of the run that produced the point file and the title of that run. The user may choose a scaling factor, and major and minor tick mark intervals, for each axis. He may optionally place a grid on the graph, and may magnify any portion of the graph. The program displays a summary of point file data, including the maximum and minimum values for each coordinate, at the beginning of each run. The user may change his selection of graph parameters before the graph is drawn, and may retain the selection of graph parameters for use with another point file. This program must be linked to subroutines in the library SYS:HGL. Those subroutines are not documented in this report.

DAT

FILM

5-8